CYPR-CD01177M

UNITED STATES PATENT APPLICATION FOR

DESIGN SYSTEM PROVIDING AUTOMATIC SOURCE CODE GENERATION FOR PERSONALIZATION AND PARAMETERIZATION OF USER MODULES

Inventor:

Kenneth Y. Ogami

prepared by:

WAGNER, MURABITO & HAO LLP

Two North Market Street

Third Floor

San Jose, California 95113

(408) 938-9060

DESIGN SYSTEM PROVIDING AUTOMATIC SOURCE CODE GENERATION FOR PERSONALIZATION AND PARAMETERIZATION OF USER MODULES

BACKGROUND OF THE INVENTION

RELATED APPLICATIONS

US Patent Application entitled, "Microcontroller Programmable System on a Chip," having attorney docket number CYPR-CD00232, filed on October 22, 2001, inventor Warren Snyder, is hereby incorporated herein by reference.

FIELD OF THE INVENTION

The present invention relates to the field of programming and configuring programmable electronic devices. More particularly, the present invention relates to using a design system for (i) displaying choices for user-selected functions; (2) displaying choices for where and how to implement the selected function; and (3) automatically generating computer code, that when executed, will configure the electronic device to perform the selected function.

RELATED ART

A microcontroller is an integrated circuit device ("chip") composed of functional units or "blocks," input/output pins, internal busses and programmable interconnections among these components. Many of these configurable components can be programmed to perform a specific function or connect to another specific sub-system by loading a particular bit pattern into a

particular register in the actual chip. Recently, a microcontroller having programmable analog and digital blocks has been introduced.

In addition, a microcontroller also has a central processing unit ("CPU") and a memory system for storing data and instructions. The CPU and the memory system can interact with a programmable block, e.g., a configurable component, by reading and writing registers associated with that component. The configurable blocks are therefore programmed by setting their configuration registers with certain values. Thus, by writing a program for the CPU, a user can specify the function the configurable component will perform as well as the connections among configurable components. The user can also write programs that interact with the configurable components once those components have been initialized. The programs also do this by reading and writing registers associated with the configurable components.

Configuring and programming a microcontroller requires specifying and setting an enormous number of bits. It is common for engineers who need to configure and program a microcontroller to develop and use software for that purpose such as design tools, databases, assemblers, compilers, linkers, and debuggers. However, conventional microcontroller programming and configuring requires that the engineer incorporate the literal binary codes and addresses associated with the configurable components into the microcontroller source code.

This conventional technique therefore requires many manual steps and is error-prone and tedious. In other words, if a designer wants to program a configurable block to implement a circuit, then the designer manually determines the configuration registers of the hardware resource. The circuit designer then manually develops code to program those registers in such a way to realize the circuit (e.g., an amplifier). The designer then has to manually write code to operate the circuit in the desired fashion, e.g., give the amplifier a specific gain, etc. When dealing with physical addresses of configuration registers and the specific configuration data needed to program them, the data and information are often expressed as a series of numbers in decimal and/or hexadecimal and/or binary formats. Unfortunately, these numbers and number formats are very complex and hard to read and remember. Any small error in syntax, or a typographical error, or a transposition error can be fatal for the overall program. Manual programming and mapping leads to such errors.

SUMMARY OF THE INVENTION

Accordingly, what is needed is a design system for automatically generating source code that incorporates configuration information for the programming of hardware resources to implement and program circuits. What is needed is such a system that can be used to program programmable electronic devices, e.g., microcontrollers. What is needed yet is a system and method for automatically generating source code to program hardware resources once a designer selects a desired circuit design and a hardware resource to implement that circuit design. The present invention provides these advantages and others not specifically mentioned above but described in sections to follow.

A method and system of automatically generating assembly code (or other source code) for configuring a programmable microcontroller are presented. Source code files are automatically generated for: (1) realizing the user module in a hardware resource; and also (2) to configure the user module to behave in a prescribed manner.

The method involves displaying virtual blocks in a computerized design system where the virtual blocks correspond to programmable circuit blocks in a programmable electronic device, e.g., a microcontroller chip. The user selects a user module (e.g., a circuit design) that defines a particular function to be performed on the microcontroller. The user module is represented, in

part, by XML data which defines the way in which configuration registers need to be programmed in order to implement the circuit design thereon. The user then assigns or allocates virtual blocks to the user module, e.g., "placement of the user module." The programmable hardware resources include both programmable analog blocks and programmable digital blocks. A user module may span multiple blocks. The hardware resources are also represented, in part, by XML data defining the physical addresses of the configuration registers.

The design system then automatically generates source code for configuring the programmable blocks to perform the desired function. This process involves a mapping of the XML data from the user module and from the selected hardware resources. The source code can then be assembled, linked and loaded into the microcontroller's memory system. When executed on the microcontroller, the executable code will then set configuration registers within the blocks to implement the function.

Two different types of configuration are discussed herein with respect to the user module. Automatic generation of source code for "personalization" of a user module is the process (performed by embodiments of the present invention) of generating source code that sets configuration registers to implement or "realize" the circuit design in the hardware resource, e.g., to implement an amplifier circuit. On the other hand, automatic

generation of source code for "parameterization" of a user module is the process (performed by embodiments of the present invention) of generating source code that sets configuration registers to cause an already implemented circuit design to behave in particular way, or to adopt some characteristic, e.g., to program the amplifier with a particular gain, etc., or to make some clocked digital circuit level sensitive or edge sensitive, etc.

More specifically, an embodiment of the present invention is directed to a method (and computer system) for generating assembly code to configure a microcontroller with programmable circuit blocks, the method comprising: displaying a collection of virtual blocks in a design system with each virtual block in the collection corresponding to a programmable block in the microcontroller; selecting a user module defining a function; assigning a virtual block taken from the collection to the user module; and automatically constructing a source code table file comprising configuration information for a programmable block of the microcontroller corresponding to the virtual block wherein the configuration information is used to cause the programmable block to implement the function.

Embodiments include the above and wherein the function can be a timer, a counter, an analog-to-digital converter, a digital-to-analog converter, a pulse-width modulator, a signal amplifier, or a serial communication unit.

Embodiments include the above and wherein the design system computes a register address for the programmable block corresponding to the virtual block assigned to the user module, devising a symbolic name for that register address and placing the symbolic name into a table file, an include file or an assembly code file.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a general purpose computer system on which embodiments of the present invention may be implemented.

Figure 2 illustrates the general architecture of a microcontroller with programmable blocks in accordance with an embodiment of the present invention.

Figure 3 illustrates the relationship between a microcontroller, a system for programming a microcontroller, and a computer system for running a design system in accordance with an embodiment of the present invention.

Figure 4 is a flow chart of steps of using a design system to automatically generate code in accordance with an embodiment of the present invention.

Figure 5 is a screen display showing the user module choices in accordance with an embodiment of the present invention.

Figure 6 is a screen display showing the assignment of a virtual block to a user module in accordance with an embodiment of the present invention.

Figure 7 is a screen display showing the assignment of two virtual blocks to a user module in accordance with an embodiment of the present invention.

Figure 8A is a data flow diagram showing the flow of information through the source generation process.

Figure 8B is a flow chart showing the construction of source code files in accordance with an embodiment of the present invention.

Figure 9 shows a portion of a template file with a generic name in accordance with an embodiment of the present invention.

Figure 10 is a screen display showing a generated assembly code file in accordance with an embodiment of the present invention.

Figure 11 illustrates the relationship of different code files to the design system and a microcontroller in accordance with an embodiment of the present invention.

Figure 12 illustrates the relationship of an executing program to the programmable blocks in accordance with an embodiment of the present invention.

Figure 13 shows a portion of an assembly code table file in accordance with an embodiment of the present invention.

Figure 14 shows the use of a symbolic name as a register address in assembly code in accordance with an embodiment of the present invention.

Figure 15 shows a portion of an assembly include file defining the symbolic name in accordance with an embodiment of the present invention.

Figure 16 shows a portion of a generated exemplary C-header file defining procedural interfaces for managing a counter in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

In the following detailed description of the present invention, a design system providing automatic source code generation for personalization and parameterization of user modules, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be obvious to one skilled in the art that the present invention may be practiced without these specific details. In other instances well known methods, procedures, components, and circuits have not been described in detail as not to unnecessarily obscure aspects of the present invention.

NOTATION AND NOMENCLATURE

Some portions of the detailed descriptions that follow are presented in terms of procedures, steps, logic blocks, processing, and other symbolic representations of operations on data bits that can be performed on computer memory. These descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. A procedure, computer executed step, logic block, process, etc., is here, and generally, conceived to be a self-consistent sequence of steps or instructions leading to a desired result. The steps are those utilizing physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated in a computer system. It has proven convenient at

times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussions, it is appreciated that throughout the present invention, discussions utilizing terms such as "checking," "comparing," "accessing," "processing," "computing," "suspending," "resuming," "translating," "calculating," "determining," "scrolling," "displaying," "recognizing," "executing," or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

COMPUTER SYSTEM 112

Aspects of the present invention, a design system for automatically generating assembly code to configure a microcontroller, are discussed in terms of steps executed on a computer system. Although a variety of different computer systems can be used with the present invention, an exemplary computer system 112 is shown in Figure 1.

Exemplary computer system 112 comprises an address/data bus 100 for communicating information, a central processor 101 coupled with the bus for processing information and instructions, a volatile memory 102 (e.g., random access memory) coupled with the bus 100 for storing information and instructions for the central processor 101 and a non-volatile memory 103 (e.g., read only memory) coupled with the bus 100 for storing static information and instructions for the processor 101. Computer system 112 also includes a data storage device 104 ("disk subsystem") such as a magnetic or optical disk and disk drive coupled with the bus 100 for storing information and instructions and a display device 105 coupled to the bus 100 for displaying information to the computer user.

Also included in computer system 112 is an alphanumeric input device 106 including alphanumeric and function keys coupled to the bus 100 for communicating information and command selections to the central processor 101. Generally, alphanumeric input device 106 is called a keyboard or keypad. System 112 also includes a cursor control or directing device 107 coupled to the bus for communicating user input information and command selections to the central processor 101. Within the context of the present invention, the cursor directing device 107 can include a number of implementations including a mouse device, for example, a trackball device, a joystick, a finger pad (track pad), an electronic stylus, an optical beam directing device with optical receiver pad, an optical tracking device able to track the movement of a user's finger,

etc., or any other device having a primary purpose of moving a displayed cursor across a display screen based on user displacements.

Computer system 112 of Figure 1 can also include an optional signal generating device 108 coupled to the bus 100 for interfacing with other networked computer systems, e.g., over the Internet. The display device 105 of Figure 1 utilized with the computer system 112 of the present invention may be a liquid crystal device, other flat panel display, cathode ray tube, or other display device suitable for creating graphic images and alphanumeric characters recognizable to the user.

DESIGN SYSTEM PROVIDING AUTOMATIC SOURCE CODE GENERATION FOR PERSONALIZATION AND PARAMETERIZATION OF USER MODULES IN ACCORDANCE WITH EMBODIMENTS OF THE PRESENT INVENTION

In a preferred embodiment, a computer implemented design system provides the capability of automatically generating source code that, when compiled or assembled, linked and loaded, initializes, specifies and controls configurable elements within a programmable electronic device, e.g., a microcontroller to perform a user-selected function.

In a preferred embodiment of a microcontroller, some of the configurable elements are grouped into programmable blocks ("blocks"). The blocks can be grouped into families. Every block in the same family is substantially analogous. In a preferred embodiment, there is a family of blocks supporting

analog functions and a family of blocks supporting digital functions. Each block has one or more registers. Each block can be programmed to perform different functions and connect to other blocks by specifying values for each configuration register in the block. This initial specification of the components is typically performed as soon as the microcontroller "boots-up". This is referred to as "personalization" of the microcontroller to realize a user module design in selected hardware resources. In addition, a block can be configured to have registers that are used to adjust, control or measure the performance of the personalized block. This is referred to as "parameterization."

US Patent Application entitled, "Microcontroller Programmable System on a Chip," having attorney docket number CYPR-CD00232, filed on October 22, 2001, inventor Warren Snyder, is hereby incorporated herein by reference and describes such a programmable microcontroller.

Figure 2 shows a generic architecture of a microcontroller 200 with a family of programmable blocks 220, 221, 222, 223, 224, 225, also called "hardware resources." The microcontroller 200 has a microcontroller CPU 210 connected to a controller memory system 205 via a bus 215. The bus 215 also allows data from outside of microcontroller 200 to come into the controller memory system 205 via interface 240. The microcontroller CPU 210 defines the function performed by each block by setting the contents of configuration registers associated within each block by control bus 230. Control bus 230 may include logic. Each configuration register in the microcontroller has a

distinct physical register address. By executing a "load register" instruction from controller memory system 205 register with a register address for a particular block, that block will be configured in accordance with the contents loaded. In some cases, to completely configure a block or to interconnect that block to other resources in the microcontroller 200, it may be necessary to load several configuration registers. The configuration registers may be located in a separate memory space within the microcontroller 200.

Figure 3 shows the overall relationship of a design system 310 operating in a computer system 112 with a microcontroller 200 installed in controller programming system 300. The controller programming system 300 is connected to the computer system 112 by a communications link 305. Communications link 305 could be a serial line, a parallel line, a network connection or other comparable inter-computer communication mechanism. Controller programming system 300 could include in-circuit emulation capabilities. It is appreciated that the design system 310 can be used without the microcontroller 200 attached thereto in order to determine the information required to program the chip. However, when the final image is determined and it is time to program the device, the microcontroller needs to be attached to the programming device.

In general, the design system 310 models the actual configurable elements and blocks found in microcontroller 200 with corresponding virtual elements and blocks. In the discussion that follows, the term "block" as used in

context of a design system means "virtual block," while the same term used in the context of a microcontroller means an actual block or "hardware resource."

Figure 4 shows a computer implemented flow chart for using a design system to automatically generate source code for both parameterization and personalization of a user rmodule. In step 450, the user selects the desired functions in the form of user modules. Generally, a user module can be viewed as an integrated circuit design, e.g., an amplifier, a counter, a timer, a modem, etc. The user can select multiple instances of the same type of user module. In step 460, the user assigns blocks to a user module, e.g., "places" the user module. This step is repeated for each user module selected in step 450. In step 470, after assigning blocks, the design system then can automatically generate the assembly code for configuring the actual blocks.

More specifically, a user module is the collection of information necessary to implement a particular function using one or more generic blocks. This would include the specific values that need to be loaded into a block's registers to implement the circuit design. The design system can load the information about user modules from a file when the design system is initialized. In one embodiment, the information necessary to represent a user module is formated using XML data.

Figure 5 shows an exemplary screen display from a design system to allow the user to select user modules. The user modules are grouped

according to function. In a preferred embodiment, the functions can include, for example, analog-to-digital converters 410, counters 411, digital to analog converters 412, amplifiers 413, pulse width modulators 414, serial communication units 415, and timers 416, etc. Almost any circuit design can be represented as a user module. The functions can also come in different sizes or capabilities, such as 8-bit counters or 16-bit counters, etc. An 8-bit counter user module 420 is being examined. A data sheet 430 describing the details of the examined user module is also displayed. The user can double click on the icon for examined user module to add it to a collection of selected user modules.

Figure 6 shows an exemplary screen display from the design system for managing step 460. The collection of selected user modules selected is displayed in window 501. A second window 502 displays the available virtual blocks and other configurable elements. In one embodiment, there are analog blocks 530 and digital blocks 525, input bus 520, and output bus 521. The user places the user module (or, equivalently, assigns blocks to a particular user module) by choosing a specific user module 500 from the collection of selected user modules, and then pushing an advance placement button 503 to move through different candidate placements. Once a desired placement is found, the user then pushes a commit button 504. In this example, the user module only requires one digital block, which has been assigned to a specific virtual block 525. In some cases, a user module will require multiple blocks. A user module may use more than one block and may use both digital and analog

blocks. Parameter window 510 displays additional user-specifiable information about the selected user module 500. It is appreciated that this information is used for parameterization of the user module. After assigning virtual blocks to all selected user modules, the user then presses a generate code button 507 to generate files in step 470.

Figure 7 shows a screen display for a situation in which a user module uses more than one block (e.g., digital blocks 531 and 532) and is shown for a selected digital-to-analog converter 506. Parameter window 533 displays additional user-specifiable information about the selected user module 506. It is appreciated that this information is used for parameterization of the user module.

Figure 8A shows the flow of information used in generating code in step 470. The code generation module 800 implements step 470. It obtains the details of how to generally configure a particular block to perform a specific function from user module definitions 805. The code generation module 800 obtains information about which virtual block to use for which user module and specific settings within a user module from the personalization and parameterization information 810 that is provided from the graphical user interface in the design system. The template files 815 are used to provide a starting point for building the code. Symbolic names derived from the user module definition 805 and the personalization and parameterization information 810 are substituted for generic names in copies of the template files 815. The

code generation module 800 constructs source code files for personalizing the micro-controller 820 and source code files for parameterizing the micro-controller 825.

Figure 8B shows a computer implemented flow chart providing more details for step 470. In step 700, the design system defines symbolic names for the register addresses for the assigned blocks in step 460. Symbolic names are also derived for the addresses of the various assembly routines that will be used to configure the actual blocks to implement the selected user modules. In one embodiment, the symbolic name is derived from the type of user module, the instance name, and the function associated with that register or routine. The register address for a particular function for a particular instance of a user module is determined from the block assigned to that instance of the user module and the register within a generic block specified by the user module for that particular function.

In step 710, one or more assembly code files are generated for each instance of a user module. Each file is constructed from a data file, e.g, a template file, by substituting the symbolic names constructed in step 700 for generic names in the template file. A sample portion of a template file is shown in Figure 9 with a generic name 633.

In step 720, one or more assembly code files are generated that refer to all of the user modules ("global files"). These can include assembly code table

files, include files, header files, as well as interrupt service routine code. These files can also be generated from a template file.

Figure 10 shows a screen display with a window 600 showing the various generated files. Window 611 displays the contents of a selected file 601.

Figure 11 shows the relationship of template files, assembly code files, executable code files, and downloaded code to the design system 310 operating on a computer system 112 and a controller programming system 300. In general terms, the design system will read template files 320, and produce assembly, include and header files 325. The design system will compile, assemble and link the files together to produce an executable file 330. The design system then can download the executable file 330 to the microcontroller programming system 300, which in turn places it in the microcontroller memory system 205 as downloaded code block 340. As shown in Figure 12, the CPU 210 executes code block 340 to configure various programmable blocks 220, 221.

Figure 13 shows a portion of a generated assembly code table file where the configuration contents of the various registers is placed. The symbolic name 632 associated with a register is in a comment.

Figure 14 shows a portion of a generated assembly code file where the symbolic name is used to specify the register that is being loaded with a new value.

Figure 15 shows a portion of a generated assembly include file that defines the values associated with the generated symbolic names.

Figure 16 shows a portion of a generated C language header file. Generated header files would allow a user to develop code in C that could manipulate the various registers specified by the user.

The preferred embodiment of the present invention, a system and method for automatically generating assembly code to configure programmable elements of a microcontroller, is thus described. While the present invention has been described in particular embodiments, it should be appreciated that the present invention should not be construed as limited by such embodiments, but rather construed according to the below claims.